1973

Integer Programming by Cutting Planes Methods

Sung-Yen Wu
Utah State University

INTEGER PROGRAMMING BY CUTTING PLANES METHODS

by

Sung-Yen Wu

A report submitted in partial fulfillment
of the requirements for the degree

of

MASTER OF SCIENCE

in

Applied Statistics

Plan B

Approved:

UTAH STATE UNIVERSITY
Logan, Utah

1973

ACKNOWLEDGMENTS

I wish to express my sincere thanks to Dr. Bartell C. Jensen for his valuable suggestions and guidance in the preparation of this report. I would also like to thank Professor Ronald V. Canfield and Dr. Grace Y. Chen for their time spent and support given in this effort.

A great deal of appreciation and thanks go to my parents for their support in my graduate studies at Utah State University.

Sung-Yen Wu

TABLE OF CONTENTS

	Page
ACKNOWLEDGMENTS	ii
LIST OF TABLES	iv
LIST OF FIGURES	v
INTRODUCTION	1
BASIC CONCEPT OF LINEAR PROGRAMMING	5
Description	5
Slack, surplus and artificial variables	6
The simplex method	7
The dual simplex algorithm	13
INTEGER PROGRAMMING BY CUTTING PLANES METHODS	14
Description	14
Formulation methods and solutions	15
NUMERICAL EXAMPLES	23
COMPARISON OF THE EFFECIENCY, USAGE, AND RELATIONSHIP OF DIFFERENT CUTTING PLANES METHODS	34
Summary	35
REFERENCES	37
APPENDIX	38
Computer Programs	39

LIST OF TABLES

Table		Page
1.	The initial primal feasible tableau	23
2.	The optimal linear programming tableau with no cut	24
3.	The optimal tableau after the first Gomory's cut . .	25
4.	Computation of D_5^5	27
5.	Computation of D_5^3	28
6.	Computation of D_5^1	28
7.	The optimal tableau after the first Kianfer's cut . .	29
8a.	Gomory's algorithm	36
8b.	Kianfar's algorithm	36

LIST OF FIGURES

Figure Page

1. Cutting Planes Methods on two dimensions 4

INTRODUCTION

Linear programming is a relatively new, very important branch of modern mathematics and is about twenty five years old.

In this day and age, most planners and decision makers will acknowledge that some linear optimization problems are worth the expense and trouble to solve. Using linear programming technique as a tool to make decision plannes are able to greatly redice cost or increase profit for any project under consideration.

Since Dr. George B. Dantzig published his first paper on the simplex method in 1947, progress in that field has been rapid. Although the first applications were military in nature, it was not long before it became apparent that there were important economic and industrial applications as well.

Typical areas of application in linear programming are inventory control, quality control, production scheduling, forecasting, maintenance and repairs, process design, accounting procedures, capital budgeting, the diet problem, machine-loading problem, and the transportation problem.

Integer linear programming was first introduced by Dantzig, Fulkerson, and Johnson in 1954 in their work on solving the traveling salesman problem and subsequently by Markowitz and Manne in

their work on the solution of discrete programming problems. In 1958 Gomory developed a systematic procedure for obtaining cuts which can be shown to lead to an optimal solution in a finite number of steps, when all of the variables are constrained to be integers. Later, in 1960, the algorithm for solving the mixed-integer case was also developed.

In many practical problems, some or all the variables are restricted to integer values. Integer programming was introduced to solve such kind of problems. Integer programming can be used to solve such problems as the fixed-charge problem, discrete programming problems, project planning, manpower scheduling, and the traveling salesman type problems as well as capital budgeting problems. The most interesting integer programming problems are those with variables taking only one of the values 0 or 1.

At present, many algorithms are available for solving linear programs of the type discussed here. Four of the most famous methods are:

1. Cutting-plane algorithms (Gomory, Balas, Kianfar)
2. Group theory for the all integer problem (Gomory, Shapiro)
3. Decomposition (Benders)
4. Enumerative
    a. Implicit enumeration for the pure integer problem (Balas)

b. Branch and Bound (Land and Doig, Litter, Dakin, Tomlin).

The intention of this study is to review some of the cutting plane methods and to compare one with another in terms of the number of interations required and the computation time which results from each method.

There are two methods that will be considered. The first is Gomory's method which adds a new constraint, one at a time, to the original constraint system to yield a sequence of new linear programming problems until the optimal solution has satisfied the integerality requirement (Hadley, 1964).

The second method is Kianfar's stronger inequalities for 0,1 integer programming using Knapsack Functions (Kianfar, 1971).

In Gomory's method, the integer points outside the 0,1 space can limit the parallel movement of the hyperplane of the cut toward the solution set, furthermore it does not allow this hyperplane to rotate for a deeper cut. This method removes these two limitations to form a stronger cut in solving 0,1 integer programming problem. The cut means to add a new constraint to the original linear programming problem such that some areas in the convex set are cut off. (See Figure 1)

This report surveys the literature on basic linear programming, the simplex method and dual simplex algorithm used in integer

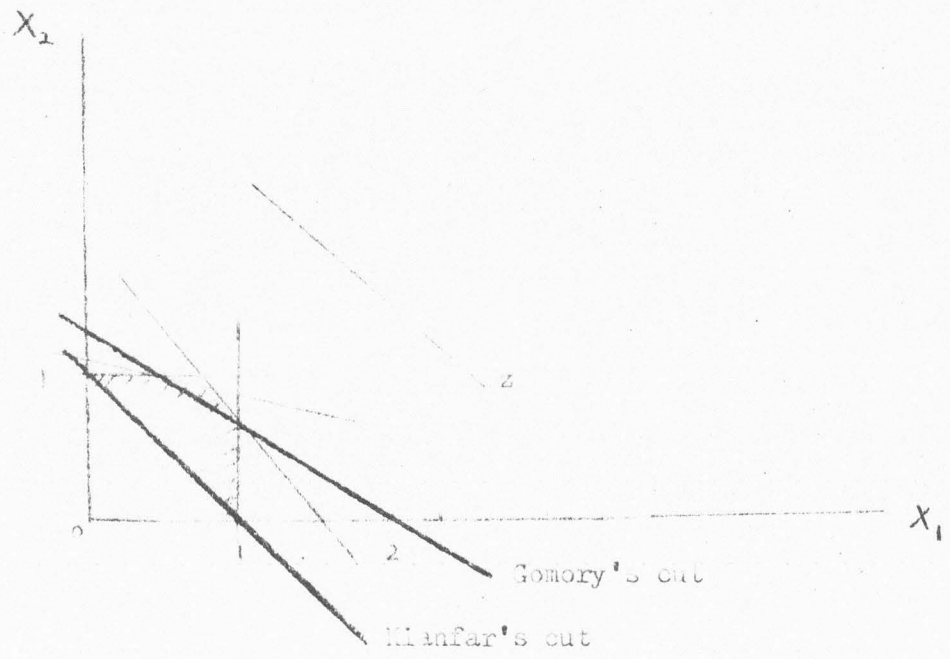

Figure 1. Cutting Planes Methods on two dimensions

programming. It also illustrates two methods of finding solutions to 0, 1 integer linear programming problem by cutting planes methods.

Finally, computer programs will be written for the two methods to compare their efficiency one with the other. Efficiency as defined here is the time required on an electronic computer or the number of iterations and arithmetic operations. Moreover, an efficient algorithm may be one which requires relatively little computer core storage.

BASIC CONCEPT OF LINEAR PROGRAMMING

Description

The goal in solving a linear programming problem is to identify an optimal solution and its associated value of the objective function.

Mathematically speaking, we have m inequalities or equations in r variables (m and r are independent) of the form:

$$\text{max. or min. } Z = c_1 x_1 + c_2 x_2 + \ldots + c_r x_r \qquad (1)$$

subject to the constraints

$$a_{i1} x_1 + a_{i2} x_2 + \ldots + a_{ir} x_r \left\{ \begin{matrix} \leq \\ = \\ \geq \end{matrix} \right\} b_i, \quad i = 1, \ldots, m \qquad (2)$$

where the a_{ij}, c_j and b_i are assumed known constants, and (1) is called the objective function. Usually, some or all of the variables are restricted to be non-negative. Sometimes, it may be required that some or all of the variables are allowed to take only integral values.

When (1) and (2) are used as a model for a decision problem, the variables (x_i, $i = 1, \ldots, r$) represent the decision maker's possible policy choices, the relations (2) express the constraints, and the objective function is the measure of effectiveness.

Any set of x_j which satisfies the set of constraints (2) is called a solution. Any solution which satisfies the non-negativity restrictions is called a feasible solution. Any feasible solution which maximizes the z of (2) is called an optimal feasible solution.

The problem itself may be infeasible, if no solution can satisfy every constraint. Another possibility is that the problem has no finite maximum value of objective function Z. Such a problem is unbounded or has an unbounded optimal solution (Hadley, 1962).

Slack, surplus and artificial variables

We note that an inequality of the type

$$\sum_{j=1}^{r} a_{hj}x_j \leq b_h \qquad (3)$$

can be converted into an equation by adding a slack variable x_{r+h}, where $x_{r+h} \geq 0$.

$$\sum_{j=1}^{r} a_{hj}x_j + x_{r+h} = b_h \qquad (4)$$

Next consider inequalities having a $\geq$ sign

$$\sum_{j=1}^{r} a_{kj}x_j \geq b_k \qquad (5)$$

Equation (5) can be rearranged to

$$\sum_{j=1}^{r} a_{kj} x_j - x_{r+k} = b_k \qquad (6)$$

by adding a surplus variable x_{r+k}, where $x_{r+k} \geq 0$.

The effect of this change is that the inequality constraints with $\leq$ or $\geq$ signs have been converted into equations, so that a system of simultaneous linear equations is obtained. The adjusted problem has the same set of optimal solutions as the original one (Hadley, 1962).

Usually it starts in initial basic solution by setting the basis matrix B = I.

If some constraints do not require the addition of a slack variable, the initial basic feasible solution may not be easy to find.

In such cases, artificial variables and vectors are added to the set of constraints to ensure an initial basic feasible solution to a linear programming problem can be found.

The simplex method

In solving an integer linear programming problem by the use of cutting plans, the simplex method seems more suitable and common than the revised simplex method. The reason is in the simplex method, it transforms all the y_j, x_B, $z_j - c_j$ and z at each iteration.

Where $y_j = B^{-1} a_j$, $j = 1, \ldots, n$, $x_B = B^{-1} b$ $\quad c_B = (c_{B1}, \ldots, c_{Bm})$,

$z = c_B x_B$, $z_j = c_B y_j$, the basis matrix $B = (b_1, \ldots, b_m)$. All such

information in the simplex method tableau is needed to form a cut in integer programming.

Although the revised simplex method has the advantage of speed and storage saving over the standard simplex method, it does not yield all the information to form a cut. This is due to the fact that only the basis inverse is transformed at each iteration.

The simplex method is a scheme which systematically evaluates the extreme points of the convex set in terms of the objective function until a terminal status is reached.

Three types of terminal status can be identified.

(a) a finite optimal solution; that is an optimal basic feasible solution.

(b) an infinite optimal solution; that is an unbounded solution.

(c) the problem has no feasible solution; that is the constraints can not simultaneously be satisifed.

The simplex method starts with a basis that yields a basic feasible solution. It then moves on to a new basis associated with a better basic feasible solution which increases the value of the maximizing function (or vis vasa) by means of replacing one of the current basis vectors with a non-basis vector. The non-basic vector to enter the basis will yield the greatest contribution to the improvement of the objective function if it is a maximizing function. On the other hand, the current basis vector to be removed is selected according to a rule which insures the continued feasibility of the new solution.

After the vectors to enter and leave the basis have been found, we start to form a new basis by using transformation formulas. Once the new basis is established, it forms the new starting point for repeating the same evaluative process to determine a still better basic feasible solution. This continues until the final stage is reached.

Computational procedure for the simplex mothod (Hadley, 1962).

Step 1. One wishes to solve the linear programming problem

$$\sum_{j=1}^{r} a_{ij} x_j \left\{ \leq \; = \; \geq \right\} b_i, \quad i=1,\ldots,m \tag{7}$$

to maximize the linear objective function

$$Z = \sum_{j=1}^{r} c_j x_j \tag{8}$$

with the non-negativity restrictions

$$x_j \geq 0, \quad (j=1,\ldots,r)$$

For the convenience of computation, at the beginning check to see that all the b_i are non-negative. If any of the b_i are negative multiply it by -1 to obtain $b_i \geq 0$. Next add slack or surplus variables to the inequality constraints

Then

$$\sum_{j=1}^{r} a_{ij} x_j \leq b_i$$

will bcome

$$\sum_{j=1}^{r} a_{ij} x_j + x_{r+i} = b_i, \; x_{r+i} \geq 0 \qquad (9)$$

and

$$\sum_{j=1}^{r} a_{ij} x_j \geq b_i$$

will become

$$\sum_{j=1}^{r} a_{ij} x_j - x_{r+i} = b_i, \; x_{r+i} \geq 0 \qquad (10)$$

every slack or surplus variable is given a price zero. After the conversion the problem will become

$$Ax = b, \; x \geq 0, \quad \max z = cx$$

where

$$A = (a_1, \ldots, a_n).$$

Step 2. Check to see whether the matrix A contains an m x m identity matrix, assumes r(A) = m. If the identity matrix does not exist, add the necessary number of artificial variables and artificial vectors to obtain an identity matrix. The two phase method is used to treat artificial vectors. The phase I drive all artificial variables to zero and the phase II maximize the actual objective function z.

In phase I, assign prices of zero to the legitimate variables

and of -1 to the artificial variables. Now maximize the function

$$z = \sum_{i=1}^{s} (-1)x_{ai} = -1X_a \qquad (11)$$

where X_a is an s-component column vector containing the artificial variables. Notice, that $z^* \leq 0$ is always true, and the maximum value of z^* is zero.

At the end of phase I, we may have max $z^* < 0$. This means the problem has no feasible solution. If we have max $z^* = 0$ at the end of phase I, we have a basic feasible solution to the original problem, and all the artificial variables have been driven to zero.

In phase II, we assign its actual price c_j to each legitimate variable and a price of zero to any artificial variables which may appear in the basis at a zero level. The function to be optimized is the actual objective function z.

Step 3. Construct an initial tableau, and compute $z_j - c_j$ from

$$z = c_B b, \quad z_j - c_j = c_B a_j - c_j \qquad (12)$$

Step 4. If all $z_j - c_j \geq 0$, there are two possibilities. If some of the x_B in the basis do not satisfy the non-negativity restrictions we have an infeasible solution, otherwise an optimal basic feasible solution is obtained.

If one or more $(z_j - c_j) < 0$, proceed by determining the vector a_k in A but not in B to enter the basis from

$$z_k - c_k = \min_j (z_j - c_j), \quad z_j - c_j < 0 \tag{13}$$

After a_k is selected, check y_{ik}. if $y_{ik} \leq 0$ for all i, there have an unbounded solution.

Step 5. If $y_{ik} > 0$ for at least one i, proceed by determining the vector to be removed from the basis. The criteria that insures feasibility is

$$\frac{x_{Br}}{y_{rk}} = \min_i \left\{ \frac{x_{Bi}}{y_{ik}}, \; y_{ik} > 0 \right\} \tag{14}$$

where the subscript y identifies the vector to leave the basis.

Step 6. Use the vector transformation formuli to find the new basic feasible solution. The "ring around the rosy" method is

$$\hat{y}_{ij} = y_{ij} - \frac{y_{ik}}{y_{rk}} y_{rj}; \quad \text{all } j, \; i=1,\ldots,m+1, \; i \neq j$$

$$\hat{y}_{rj} = \frac{y_{rj}}{y_{rk}}, \quad \text{all } j \tag{15}$$

$$x_B = y_0, \; z = y_{m+1,0}, \; z_j - c_j = y_{m+1,j}, \; j=1,\ldots,n$$

Step 7. Go to step 4, for another iteration. There will be a finite iterative process to find an optimal solution.

The dual simplex algorithm

Every linear programming problem (primal) has associated with it a companion problem which is called the dual.

The solution to the dual is simultaneously determined when solving the primal problem. In many cases, the duality can be of considerable help in solving linear programming problems.

Because the dual simplex algorithm will be used to determine the vectors to enter and leave basis in integer linear programming, a short discussion of this algorithm will be presented here.

When the simplex tableau have one or more $x_{Bi} < 0$, but with $z_j - c_j \geq 0$ for all j, the criteria for the change of basis is;

1. Vector to be removed from basis: Choose

$$x_{Br} = \min x_{Bi}, \; x_{Bi} < 0 \tag{16}$$

column br is removed from the basis and x_{Br} is therefore driven to zero.

2. Vector to enter basis: The vector a_k to enter the basis is determined from

$$\theta = \frac{z_k - c_k}{y_{rk}} = \max_j \frac{z_j - c_j}{y_{rj}}, \; y_{rj} < 0 \tag{17}$$

Note that in the dual simplex algorithm one first determines the vector to leave the basis and then the vector to enter. This is the reverse of that done in the simplex method.

INTEGER PROGRAMMING BY CUTTING PLANES METHODS

Description

An integer programming problem is a linear programming problem with the additional feature that some or all of the variables are required to take on integer values only.

The most general problem of this type of problem can be written as

$$Ax = b;\ x \geq 0;\ x_j \text{ an integer},\ j \in J_1$$
$$\max z = cx \tag{18}$$

if J_1 contains all j, we have an all integer problem. If J_1 is empty, (18) is a linear programming problem and if J_1 contains part of j, we have an mix-integer problem. In an all integer problem if x_j can take only the value of 0 or 1, we consider it a 0, 1 integer programming problem.

In theory, every integer linear programming problem can be solved. However, in practice, due to the difficulties of too much time involved to obtain the solution (even with today's computer) and the effect of round-off error and machine storage size, more research work in this area is still needed.

Formulation methods and solutions

A basic technique for solving the integer programming problems is (Hadley, 1964).

Step 1. Solve the problem ignoring the integrality restriction.

Step 2. If an optimal solution is not feasible due to the violation of the integrality restrictions, a new constraint is added to the original constraint equations, then proceed to step 1. Otherwise, the optimal solution with the required integer variables has been obtained.

The first method discussed here is Gomory's Algorithm. The essential nature of this algorithm is that it provides the proof of finiteness.

Gomory's algorithm for the all integer programming problem

Step 1. Solve LP (18) and let the optimal solution so obtained by $x_B = y_0$, where

$$y_o = x_B + \sum_{j \in R} y_j x_j \qquad (19)$$

R is the set of j corresponding to the nonbasic variables.

Step 2. If all components of y_0 are integers, we have an optimal solution; otherwise let

$$x_{Bu} = y_{u0} - \sum y_{uj} x_j, \qquad (20)$$

where y_{u0} is not integeral.

Now write

$$y_{uj} = \sigma_{uj} + f_{uj}, \; j \in R; \; y_{u0} = \sigma_{u0} + f_{u0}, \tag{21}$$

where σ_{uj} is the largest integer $\leq y_{uj}$,

$j \in R$ and $j = 0$; and $0 \leq f_{uj} < 1$.

If more than a single component of y_0 is nonintegral, we adopt the equation for which f_{u0} is largest, where $0 < f_{u0} < 1$.

Step 3. Add constraint (22) to the constraints of LP (16)

$$f_{u0} - \sum_{j \in R} f_{uj} x_j \leq 0 \tag{22}$$

where

$$f_{u0} > 0, \; 1 > f_{uj} \geq 0.$$

Step 4. Use the dual simplex algorithm to obtain an optimal solution to the augmented problem and let $x_B = y_0$ then return to step 2.

Gomory's algorithm for the mixed-integer programming problem.

Step 1. Same as all integer case.

Step 2. Same as all integer case.

Step 3. Use formula (23) and (24) to obtain the cut.

$$\sum_{j \in R} (-d_{uj}) x_j \leq -f_{u0} \tag{23}$$

$$d_{uj} = \begin{cases} y_{uj},\ j \in R^*_+, & \text{for the variables not required to be integer.} \\ \dfrac{f_{u0}}{1-f_{u0}} \left| y_{uj} \right|,\ j \in R^*_- & (24) \\ f_{uj},\ j \in R \cap J_1,\ f_{uj} \leq f_{u0}, & \text{for the variables require to take integer value.} \\ \dfrac{f_{u0}}{1-f_{u0}} (1-f_{uj}),\ j \in R \cap J_1,\ f_{uj} > f_{u0}, & \end{cases}$$

where R^* is the subset of R not in $R \cap J_1$,

R^*_+ and R^*_- are the subsets of R^* for which

$y_{uj} \geq 0$ and $y_{uj} < 0$ respectively.

Step 4. Same as all integer cases.

Another method for solving the 0, 1 all integer programming problem is suggested by Ferydoon Kianfar. This method uses Gomory's cutting-plane method to obtain the first cut and then allows the hyperplane of that cut to be rotated in any direction until the final hyperplane passes through as many integer points not in convex set S (defined below) as possible (Kianfar, 1971).

Let the sets of points be

$$S = \left\{ X = (x_1, \ldots, x_n) \mid x_j = 0, 1,\ j = 1, \ldots, n \right\} \qquad (25)$$

and

$$A = \left\{ X = (x_1, \ldots, x_n) \mid x \in S; \sum_{j=1}^{n} a_j x_j \leq L \right\} \tag{26}$$

where a_j, all j, and L are positive integer constants.

The problem is to find the largest integer a'_r, $r \leq n$, denoted by $\hat{a}_r$, such that $a'_r > a_r$ and $A = A_r$, where

$$A_r = \left\{ X \mid X \in S; \sum_{j=1,\ j \neq r}^{n} a_j x_j + a'_r x_r \leq L \right\} \tag{27}$$

if we assume that $a_r > a_r$, then the inequality

$$\sum_{j=1,\ j \neq r}^{n} a_j x_j + \hat{a}_r x_r \leq L \tag{28}$$

is stronger than the original inequality

$$\sum_{j=1}^{n} a_j x_j \leq L \tag{29}$$

where $\hat{a}_r$ should be selected such that

$$A_1 = \left\{ X \mid X \in A;\ x_r = 1 \right\},\ A_1 \subset A_r \text{ and } A_r = A \text{ for } a'_r = a_r$$

this is equivalent to selecting $\hat{a}_r$ as

$$\hat{a}_r = L - \max_{X \in A} \sum_{j=1,\ j \neq r}^{n} a_j x_j \tag{30}$$

but in fact $\hat{a}_r$ is equal to $a_r + S_n^r (L - a_r)$ by theorem 1

where

$$S_n^r(L-a_r) = L - a_r - F_n^r(L-a_r) \tag{31}$$

and

$$F_n^r(L-a_r) = \max_{X\in A} \sum_{1j=1,\, j\neq r}^{n} a_j x_j \tag{32}$$

is a special case of the Knapsack Function (Kianfar, 1971).

Using the Knapsack Function we have

Theorem 1. let

$$M = \left\{ X = (x_1, \ldots, x_n) \,\middle|\, \sum_{j=1}^{n} a_j x_j \leq L;\ x_j = 0, 1, 2, \ldots, \text{all } j \right\} \tag{33}$$

$$N = \left\{ X = (x_1, \ldots, x_n) \,\middle|\, \sum_{j=1,\ j\neq r}^{n} a_j x_j + (a_r + k_r)x_r \leq L; \right.$$

$$\left. x_j = 0,\ 1,\ 2, \ldots,\ \text{all } j \right\}, \tag{34}$$

where

$$0 \leq k_r \leq \min_{n=1, 2, \ldots, [L/a_r]} (S_n^r(L-ha_r)/h)$$

Then M = N. (The notation [y] means the largest integer less than or equal to y.)

Proposition 1.

$$S_1(k) = \infty,\ k < 0;\ S_1(k) = k,\ 0 \leq k < a_1;$$

$$S_1(k) = k - a_1,\ a_1 \leq k \leq L; \tag{35}$$

$$S_j(k) = \min\left\{ S_{j-1}(k),\ S_{j-1}(k-a_j) \right\}\ (j=2, \ldots, n)$$

Proposition 2.

$S_j(k)$ is either zero or is equal to $S_j(k-1) + 1$.

Where $S_j(k) = k - k'$ and k' is the largest integer $p \leq k$ such that $S_j(p)=0$.

Now let

$$D_j = \left\{ k \mid S_j(k) = 0,\ 0 \leq k \leq L \right\};\ (j=1,\ldots,n),$$

then proceed to find D_n. From (35) we obtain

$$D_1 = \left\{ 0,\ a_1 \right\},\ a_1 \leq L, \tag{36}$$

$$D_j = D_{j-1} + \left\{ k+a_j \mid k \in D_{j-1},\ k+a_j \leq L \right\},\ (j=2,\ldots,n)$$

each D_j, $j=2,\ldots,n$ is computed from D_{j-1} by $\min \left\{ N(D_{j-1}),\ L-a_j \right\}$

where $N(D_{j-1})$ denotes the number of elements (Kianfar, 1971).

The $S_n^r(k)$ of the Knapsack Function is that

$$S_n^r(k) = k - F_n^r(k), \text{ where } F_n^r(k) = \max \sum_{j=1,\, j\neq r}^{n} a_j x_j$$

subject to $\sum_{j=1,\, j\neq r}^{n} a_j x_j = k,\ x_j = 0, 1,\ \text{all } j,$

then

$$\hat{a}_r = a_r + S_n^r(L-a_r) \tag{37}$$

and

$$S_n^r(k) = k - k', \quad (k=0,1,\ldots,L) \tag{38}$$

Using (37) and (38) we compute $\hat{a}'_n$. next, we set $r = n-1$ and compute D_n^{n-1} but D_j^n, $j=1,\ldots,n-2$ and hence

$$D_n^{n-1} = D_{n-2}^n + \left\{ k+\hat{a}_n \mid k \in D_{n-2}^n,\ k+a_n \leq L \right\}. \tag{39}$$

From Proposition 1 it follows that

$$D_{r+1}^r = D_{r-1}^n + \left\{ k+\hat{a}_{r+1} \mid k \in D_{r-1}^n,\ k+\hat{a}_{r+1} \leq L \right\} \tag{40}$$

$$D_j^r = D_{j-1}^r + \left\{ k+\hat{a}_j \mid k \in D_{j-1}^r,\ k+\hat{a}_j \leq L \right\}, \quad (j=r+2,\ldots,n) \tag{41}$$

The steps used to get a stronger cut is (Kianfar, 1971)

Step 1. Apply Gomory's algorithm to obtain original cut. The cut at this stage is

$$F^k T^k \leq 0 \tag{42}$$

where

$$F^k = (f_0^k, -f_1^k, -f_2^k, \ldots, -f_n^k)$$

is the vector of the constants of this cut. T^k has 1 as its first component and the nonbasic variables of the kth tableau as the rest of its components.

Also we define

$$F^0 = F^k P \tag{43}$$

where P is an (n+1) x (n+1) matrix whose inverse P^{-1} transforms A^0 into A^k as follows:

$$A^k = A^0 P^{-1} \tag{44}$$

The equivalent cut in 0,1 variables is then

$$F^0 T^0 \leq 0.$$

Step 2. Find $D_j^n = D_j$, $j = 1, \ldots, n-1$ using (36), and $D_n^n = D_{n-1}^n$. Compute $\hat{a}_r$ using (37) and (38) and set $r = n-1$, where r is is a variable that is to run over the coordinate currently being considered for an increase in its coefficient (Kianfar, 1971).

Step 3. If $r = 1$ go to step 4; also check whether $(L - a_r) \in D_{r-1}^n$. If yes, set $\hat{a}_r = a_r$; otherwise, using (40) and (41) to compute D_n^r, and use (37) and (38) to find $\hat{a}_r$. In either case, set $r = r-1$ and go to step 3.

Step 4. Set $D_2^1 = \{0, \hat{a}_2\}$. Use (41) to compute D_n^1, and use (37) and (38) to compute a_1, and we arrived at strongest inequality, the coefficient is denoted by $\bar{a}_j$ for all a_j

$$\sum_{j=1}^{n} \bar{a}_j x_j \leq L \tag{45}$$

NUMERICAL EXAMPLES

In the previous chapter, two methods to solve 0, 1 integer linear programming problem by cutting planes methods have been discussed. It will be helpful to illustrate the two methods by considering some simple numerical examples. As the first example, assume the following 0, 1 integer linear programming problem:

Example 1.

$$\text{Max. } 7x_1 - 8x_2 - 9x_3 + 4x_4 + 5x_5$$

subject to

$$\begin{aligned} 2x_1 + 4x_2 - 4x_3 + 2x_4 + 4x_5 &\leq 7 \\ x_1 + 3x_2 + 2x_3 + x_4 - x_5 &\leq 4 \\ 2x_2 - 2x_3 - x_4 + x_5 &\leq 3 \end{aligned} \tag{46}$$

$$x_j = 0 \text{ or } 1, \quad j = 1, \ldots, 5$$

Table 1. The initial primal feasible tableau.

x_B	1	x_1	x_2	x_3	x_4	x_5
Z	0	−7	8	9	−4	−5
x_6	7	2	4	−4	2	4
x_7	4	1	3	2	1	−1
x_8	3	0	2	−2	−1	1

Table 1. Continued

x_9	1	1	0	0	0	0
x_{10}	1	0	1	0	0	0
x_{11}	1	0	0	1	0	0
x_{12}	1	0	0	0	1	0
x_{13}	1	0	0	0	0	1

Table 2. The optimal linear programming tableau with no cut.

x_B	x	x_2	x_3	x_6	x_9	x_{12}
Z	14.75	13.00	4.00	1.25	4.50	1.50
x_4	1.00	0.00	0.00	0.00	0.00	1.00
x_7	2.75	4.00	1.00	0.25	−1.50	−1.50
x_8	3.25	1.00	−1.00	0.25	0.50	1.50
x_1	1.00	0.00	0.00	0.00	1.00	0.00
x_{10}	1.00	1.00	0.00	0.00	0.00	0.00
x_{11}	1.00	0.00	1.00	0.00	0.00	0.00
x_{13}	0.25	−1.00	1.00	−0.25	0.50	0.50
x_5	0.75	1.00	−1.00	0.25	−0.50	−0.50

First, to solve this problem by the regular simplex method without considering its integer requirement.

The solution is given in Table 2. Notice the optimal solution does not meet the integrality restriction.

To obtain the first cut, using the ($x_{B2} = x_7$) row in Table 1. Note that $y_{20} = 2.75$, $y_{22} = 4.00$, $y_{23} = 1.00$, $y_{26} = 0.25$, $y_{29} = -1.50$,

$y_{212} = -1.50$. We see that $f_{20} = 0.75$, $f_{22} = 0$, $f_{23} = 0$, $f_{26} = 0.25$, $f_{29} = 0.5$, $f_{212} = 0.5$

Thus by (22), the cut is

$$0.75 - 0.25x_6 - 0.5x_9 - 0.5x_{12} \leq 0 \qquad (47)$$

Annex (47) to the original problem and apply the dual simplex algorithm to solve it.

One obtains an optimal solution in two iterations. For this solution given in Table 3, all x are integers. Thus, one has obtained the optimal solution to (46). It is $x_1 = x_5 = 1$, $x_2 = x_3 = x_4 = 0$, $z = 12$.

Table 3. The optimal tableau after the first Gomory's cut

x_B	1	x_2	x_3	x_4	x_9	x_{s1}
Z	12.00	13.00	4.00	1.00	2.00	5.00
x_6	1.00	0.00	0.00	−2.00	2.00	−4.00
x_7	4.00	4.00	1.00	2.00	−2.00	1.00
x_1	1.00	0.00	0.00	0.00	1.00	0.00
x_{10}	1.00	1.00	0.00	0.00	0.00	0..00
x_{13}	0.00	−1.00	1.00	−1.00	1.00	−1.00
x_5	1.00	1.00	−1.00	1.00	−1.00	1.00
x_{12}	1.00	0.00	0.00	1.00	0.00	0.00

Next one considers to use Kianfer's stronger cut, which modify the Gomory's cut. From (47) one has

$$0.75 - 0.25x_6 - 0.5x_9 - 0.5x_{12} \leq 0$$

or

$$FT = (0.75,\ 0,\ 0,\ -0.25,\ -0.5,\ -0.5),\ [1, x_2, x_3, x_6, x_9, x_{12}] \leq 0$$

to convert this cut to a cut that contains only 0, 1 variables, one first finds

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 7 & -2 & -4 & 4 & -2 & -4 \\ 1 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & -1 & 0 \end{bmatrix}$$

Then $F^o = FP = (-2,\ 1, 1,\ -1,\ 1,\ 1)$

The equivalent cut with 0, 1 variables

$$F^oT^o = (-2, 1, 1, -1, 1, 1) \cdot [1, x_2, x_3, x_6, x_9, x_{11}] = 0$$

or

$$x_2 + x_3 - x_6 + x_9 + x_{11} \leq 2, \qquad (48)$$

We make this inequality stronger by the method of replacing inequalities;

Let $a_1 = 1$, $a_2 = 1$, $a_3 = -1$, $a_4 = 1$, $a_5 = 1$, $L = 2$.

To compute $\hat{a}_5$ we compute D_5^5 (See Table 4),

From (37), (38), it follows that $S_5^5(L - a_5) = S_5^5(1) = 1 - 1 = 0$, and $\hat{a}_5 = 1+0 = 1$. To find $\hat{a}_4$ we first check to see whether $(L-a_4) \in D_3^5$, and $L - a_4 = 2 - 1 = 1 \in D_3^5$, thus $\hat{a}_4 = a_4 = 1$. To find a_3, we check $L - a_3 = 2 - (-1) = 3 \notin D_2^5$; hence we compute D_5^3 using (40), (41) (See Table 5), from (37), (38) we have $S_5^3(L - a_3) = S_5^3(3) = 3-2 = 1$, $\hat{a}_3 = -1 + 1 = 0$. Next we compute $\hat{a}_2$. First we check $L - a_2 = 2-1 = 1 \in D_2^5$, hence $\hat{a}_2 = a_2 = 1$. One now computes D_5^1 as shown in Table 6, using (38), we have $S_5^1(L-a_1) = S_5^1(1) = 1 - 1 = 0$, Hence, from (37)

Table 4. Computation of D_5^5

D_1^5	D_2^5	D_3^5	$D_4^5 = D_5^5$
0	0	0	0
1	1	1	1
	2	2	2

Table 5. Computation of D_5^3

$D_3^3 = D_2^3 = D_2^5$	D_4^3	D_5^3
0	0	0
1	1	1
2	2	2

Table 6. Computation of D_5^1

D_2^1	D_3^1	D_4^1	D_5^1
0	0	0	0
1	1	1	1
		2	2

$\hat{a}_1 = a_1 = 1$

The modified cut now is $\bar{F}_0 = (-2, 1, 1, 0, 1, 1)$ or

$$x_2 + x_3 + x_9 + x_{11} \leq 2 \qquad (49)$$

To compute $\bar{F}$ we need p^{-1}, which can be obtained from Table 2.

$$p^{-1} = \begin{bmatrix} 1.00 & 0,00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 1.00 & 0.00 & 0.00 & 0.00 & -1.00 & 0.00 \\ 0.00 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 1.00 & 0.00 & 0.00 & 0.00 \\ 1.00 & 0.00 & 0.00 & 0.00 & 0.00 & -1.00 \\ 0.75 & -1.00 & 1.00 & -0.25 & 0.50 & 0.50 \end{bmatrix}$$

The vector $\bar{F}$ is

$$\bar{F} = \bar{F}_0 P^{-1} = (0.75,\ 0,1,\ -0.25,\ -0.50,\ -0.50)$$

and the stronger inequality in terms of nonbasic variables of the tableau of Table 2 is

$$0.75 + x_3 - 0.25x_6 - 0.5x_9 - 0.5x_{11} \leq 0$$

This row is annexed to Table 2 and the resulting tableau is made optimal by the dual simplex method.

Table 7 shows the optimal tableau. For this solution, the integer requirements are satisfied. Thus one has obtained the optimal solution to (46). It is

$$x_1 = x_5 = 1, \quad x_2 = x_3 = x_4 = 0, \quad z = 12.$$

Table 7. The optimal tableau after the first Kianfar's cut

x_B	1	x_2	x_3	x_4	x_9	x_{s1}
Z	12.00	13.00	9.00	1.00	2.00	5.00
x_6	1.00	0.00	−4.00	−2.00	2.00	−4.00
x_7	4.00	4.00	2.00	2.00	−2.00	1.00
x_8	2.00	1.00	−2.00	−2.00	1.00	−1.00
x_1	1.00	0.00	0.00	0.00	1.00	0.00
x_{10}	1.00	1.00	0.00	0.00	0.00	0.00
x_{11}	1.00	0.00	1.00	0.00	0.00	0.00
x_{13}	0.00	−1.00	0.00	−1.00	1.00	−1.00
x_5	1.00	1.00	0.00	1.00	−1.00	1.00
x_{12}	1.00	0.00	0.00	1.00	0.00	0.00

In addition to the above example, three more examples are providad below.

Example 2.

Max. $5x_1 - 7x_2 - 10x_3 + 3x_4 - x_5$

subject to

$$x_1 + 3x_2 - 5x_3 + x_4 + 4x_5 \leq 1$$
$$2x_1 + 6x_2 + 3x_3 + 2x_4 + 2x_5 \leq 4$$
$$2x_2 - 2x_3 - x_4 + x_5 \leq 2$$
$$x_j = 0 \text{ or } 1,\ j = 1, \ldots, 5$$

Using Gomory's algorithm, one has the following results.

Initial solution	Z = 5.77
1st cut	Z = 5.42
2nd cut	Z = 5.00

After two cuts one obtains the integer solution with

$x_1 = 1,\ x_2 = x_3 = x_4 = x_5 = 0,\ Z = 5.$

Using Kianfar's algorithm, one has the following results.

Initial solution	Z = 5.77
1st cut	Z = 5.00

The solution is the same as in the Gomory's algorithm, except this method only needs one cut to obtain an integer solution.

The third problem is from Kianfar's example.

Example 3.

$$\text{Max. } 15x_1 - 2x_2 + 16x_3 + 6x_4$$

subject to

$$36x_1 - 46x_2 + 18x_3 - 34x_4 \leq 28$$

$$75x_1 - 20x_2 + 30x_3 + 27x_4 \leq 93$$

$$78x_1 + 91x_2 + 69x_3 - 16x_4 \leq 122$$

$$x_j = 0 \text{ or } 1,\ j=1,\ldots,4$$

Using Gomory's algorithm the results are

Initial solution	Z = 20.69
1st cut	Z = 20.68
2nd cut	Z = 20.66
3rd cut	Z = 20.64
4th cut	Z = 20.62
5th cut	Z = 20.61
6th cut	Z = 20.59
7th cut	Z = 20.50
8th cut	Z = 20.38
9th cut	Z = 20.37
10th cut	Z = 20.35
11th cut	Z = 20.34
12th cut	Z = 20.28
13th cut	Z = 20.26
14th cut	Z = 20.22
15th cut	Z = 20.21
16th cut	Z = 20.21
17th cut	Z = 20.20
18th cut	Z = 20.19

19th cut	Z = 20.15
20th cut	Z = 19.95
21st cut	Z = 19.87
22nd cut	Z = 19.85
23rd cut	Z = 19.86
24th cut	Z = 19.85
25th cut	Z = 16.00

After 25 cuts the solution is $x_1 = x_2 = x_4 = 0.$ $x_3 = 1$, $z = 16$ and is all integers. The above tells us that in this problem Gomory's algorithm needs 25 cuts to obtain the solution.

Now, by using Kianfar's algorithm the results are:

Initial solution	Z = 20.69
1st cut	Z = 19.48
2nd cut	Z = 16.00

Here the Kianfar's algorithm uses only two cuts to reach the solution.

The last and the largest example is presented below:

Example 4.

$$10x_1 - 7x_2 + x_3 - 12x_4 + 2x_5 + 8x_6 - 3x_7 - x_8 + 5x_9 + 3x_{10} = \text{Max}$$

$$-3x_1 - 12x_2 + 8x_3 + x_4 + 7x_9 - 2x_{10} \leq 8$$

$$x_2 + 10x_3 + x_5 - x_6 + 7x_7 + x_8 \leq 13$$

$$5x_1 + 3x_2 - x_3 + 2x_8 + x_{10} \leq 6$$

$$4x_3 - 2x_4 + 5x_6 + x_7 - 9x_8 + 2x_9 \leq 8$$

$$9x_2 \quad -12x_4 + 7x_5 - 6x_6 \quad + 2x_8 + 12x_9 + 3x_{10} = \leq 12$$

$$8x_1 + 5x_2 - 2x_3 - 7x_4 + x_5 \quad - 5x_7 \quad + 10x_9 \quad \leq 16$$

$$x_j = 0 \text{ or } 1, \ (j = 1, \ldots, 10)$$

In this example, Gomory's algorithm used 88 cuts and still did not obtain the solution (Z = 24.12) while Kianfar's algorithm used only 13 cuts to get the solution.

The solution is

$$x_1 = x_5 = x_6 = x_{10} = 1$$

$$x_2 = x_3 = x_4 = x_7 = x_8 = x_9 = 0$$

$$z = 23.$$

COMPARISON OF THE EFFECIENCY, USAGE, AND RELATIONSHIP OF DIFFERENT CUTTING PLANES METHODS

All the four numerical examples cited in the previous chapter have been tested by computer codes generated by the author using Fortrain IV language on B 6700.

From the four examples discussed, it can be seen that the number of cuts needed to obtain the solution by Gomory's method are greater than or equal to Kianfar's method in 0,1 integer programming problem. Kianfar's method may be viewed as the modification of Gomory's method in actual practice. The only difference between these two methods is that after the cut the Gomory's method has been obtained, Kianfar's method changes that cut so as to make it stronger.

It is believed that the larger the problem the more efficiency the Kianfer's method can be seen. The results from Kianfer's method are very encouraging but of course this experience is insufficient for a firm judgement of the efficiency of the algorithm. However, Gomory's algorithm imposes a heavy burden on the storage system and processing time on a computer, if the problem is large. Kianfar's method tends to have the ability of lessening this burden.

Due to the fact that the computer programs and the numerical

examples given are problem specific, the comment made here can only be viewed as a suggestion or reference. In addition, the efficiency of each method is partly dependent on the nature and characters of the problem being solved.

Summary

In summary the advantage and disadvantage of the two methods will be discussed.

The Gomory's method: This method has the ability to solve all kinds of all integer and mix-integer programming problems. It is easier to learn and perform, and more suitable for small problems. Less arithmetic operations are needed to form a cut. However, if the problem turns large, the number of cuts needed to solve the problem increases rapidly.

The Kianfar's method: This method can be used to solve 0, 1 integer programming problem only. It is hard to learn and not efficiency on small problems. Usually more arithmetic operations are needed to form a cut, however if the problem turns large the number of cuts needed to solve the problem increases gently.

Another point that is worth mentioning is that the Kianfar's approach can also be applied to the original constraints. In order to maintain the finiteness property of the Kianfar's algorithm, Gomory's cut should be added to it every 50 or 100 iterations.

Table 8a. Gomory's algorithm

Problem No.	No. of variables	No. of constrnts	Virtual memory	Processor time	No. of cuts	No. of Ite.
1	5	4	1.17	0.12	1	6
2	5	4	1.34	0.14	2	5
3	4	4	1.64	0.22	25	48
4 **	10	7	41.22	3.11	88	299

Table 8b. Kianfar's algorithm

Problem No.	No of variables	No. of constrnts	Virtual memory	Processor time	No. of cuts	No. of Ite.
1	5	4	2.69	0.25	1	6
2	5	4	2.79	0.24	1	4
3	4	4	2.58	0.25	2	8
4	10	7	3.59	0.42	13	50

* The unit of virtual memory is kiloword-minutes.

* The unit of processor time is minutes.

* Both compilation time and execution time are all included in memory and time calculation.

**Problem No. 4 by Gomory's algorithm after 88 cuts still does not obtain the answer.

REFERENCES

Balas, Egon. 1965. "An Additive Algorithm for Solving Linear Programs with Zero-one Variables," Operations Research 13: 517-546.

Balinski, M. L. 1965. "Integer Programming; Methods, Uses, Computation," Management Science, 12(3):253-313.

Geoffrion, A. M. and R. E. Marsten. 1972. "Integer Programming Algorithms; A Framework and State-of-the-Art Survey," Management Science, 18(8)461-491.

Hadley, George. 1962. "Linear Programming." Massachusetts, Addison-Wesley, 520 p.

_____, _____. 1964. "Nonlinear and Dynamic Programming." Massachusetts, Addison-Wesley, 484 p.

Kianfar, Ferydoon. 1971. "Stronger Inequalities for Integer Programming Using Knapsack Functions" Operations Research, 19:1374-1392.

Kuo, Shan Sun. 1965. "Numberical Analysis" Massachusetts, Addison-Wesley, 341 p.

Trauth, C. A. and R. E. Woolsey. 1969. "Integer Linear Programming: A Study in Computational Efficiency," Management Science 15(9):481-493.

APPENDIX

Computer Programs

```
C       THIS PROGRAM USE GOMORY'S ALGORITHM TO SOLVE AI L
C       INTEGER LINEAR
C       PROGRAMMING PROBLEM.  WHERE II = TOTAL NUMBER OF
C       ROWS.
C       LD(I)=VARIABLES NOT IN THE BASIS.
C       B(I)=THE PIVOT ROW.
C       L(I)=VARIABLES IN THE BASIS.
C       JJ=TOTAL NUMBER OF COLUMNS. S(I,J)=TABLEAU FOR
C       SIMPLEX METHOD.
        DIMENSION S(80,180), L(80), DUJ(80),LD(80), B(80)
        READ(5,4) II,JJ
    4   FORMAT(1814)
        III=II+1
C       READ IN THE MATRIX
        DO 10 I=1, III
   10   READ(5,12) (S(I,J),J=1,JJ)
   12   FORMAT(7F10.4)
C       READ IN THE SUBSCRIPT FOR THE SLACK VARIABLE
        READ(5,4) (L(I),I=2,II)
        IIII=II-1
        REGA=0.
   21   KKK=0
        REGB=0.
C       CALCULATE NEW LAST ROW
   22   DO 23 I=2,II
        IF(L(I).NE.0) GO TO 23
        DO 27 J=1,JJ
        IF(S(I,J).EQ.0.) GO TO 27
        S(III,J)=S(III,J)-S(I,J)
   27   CONTINUE
   23   CONTINUE
C       SEARCHING FOR THE MOST NEGATIVE ENTRY
        K=III
   44   JJJ=JJ-1
        IF(REGA.EQ.1.) GO TO 500
        B(K)=0.
        L(K) = 0
        DO 42 J=1,JJJ
        IF(S(K,J).GE.0.) GO TO 42
        IF(B(K),LE.S(K,J) ) GO TO 42
```

```
      B(K)=S(K, J)
      L(K)=J
   42 CONTINUE
C     TEST FOR L(K).
      REGB=1.
   45 IF(L(K)) 46, 62, 46
C     FIND OUT THE PIVOT COLUMN
   46 KJ=L(K)
      DO 120 I=2, II
      IF(S(I, KJ)) 120, 120, 121
  120 CONTINUE
      WRITE(6, 130)

  130 FORMAT(1X, 'UNBOUNDED')
      GO TO 1
C     COMPUTTING THE RATIO
  121 JK=0
      DO 50 I=2, II
      IF(S(I, KJ), LE. 0. ) GO TO 50
      X=S(I, JJ)/S(I, KJ)
      IF(JK. EQ. 0) GO TO 53
      IF( X. GE. XMIN) GO TO 50
   53 XMIN=X
      JK=I
   50 CONTINUE
   56 X=S(JK, KJ)
      L(JK)=KJ
C     CALCULATING THE NEW ROWS ABOVE THE PIVOT ROW
      DO 57 I=1, III
   57 B(I)=S(I, KJ)
      IJ=JK-I
      DO 59 I=1, IJ
      DO 59 J=1, JJ
      IF(S(JK, J)) 58, 59, 58
   58 IF (B(I)) 580, 59, 580
  580 S(I, J) = S(I, J) - B(I)*(S(JK, J)/X)
   59 CONTINUE
C     CALCULATING THE NEW ROWS BELOW THE PIVOT ROW
      IJ=JK+1
      DO 61 I=IJ, III
      DO 61 J=1, JJ
      IF (S(JK, J)) 60, 61, 60
   60 IF(B(I))600, 61, 600
  600 S(I, J)=S(I, J)-B(I)*(S(JK, J)/X)
   61 CONTINUE
C     NORMALIZATION
```

```
      DO 205 J=1,JJ
  205 S(JK,J)=S(JK,J)/X
      KKK=KKK+1
      WRITE(6,105) KKK,S(K,JJ),L(JK)
  105 FORMAT (1X,I4,6X,F15.2,10X.I4)
      GO TO 44
   62 IF (K-1) 70,70,63
   63 IJ=JJ-1
C     TEST TO SEE WHETHER ALL THE ELEMENTS ON THE
C     LAST ROW ARE CLOSE TO ZERO.
      DO 65 J=1,IJ
      IF(S(K,J)-.0001) 65,65,66
   65 CONTINUE
      WRITE (6,103)
  103 FORMAT (1X,'FEASIBLE')
      WRITE(6,101)
  101 FORMAT(1X,'ITERATION      OBJ. FUNCTION      NEW BASIC VAR.'
C     TEST TO SEE WHETHER ALL THE ELEMENTS ON THE LAST
C     ROW
      DO 140 J=1,JJ
  140 S(III),J)=0.
      K=1
      KKK=0
      IF(REGB.EQ.0) GO TO 31
      GO TO 44
   31 REGB=1.
      GO TO 500
   66 WRITE(6,6)
    6 FORMAT(1X,'INFEASIBLE')
      GO TO 1
   70 WRITE(6,8) S(1,JJ)
    8 FORMAT(////1X,'OBJ. FUNCTION,',F20.8/,1X)
C     FIND THE LARGEST S(I,JJ)
  199 FUO=0.
      DO 202 I=2,II
      REMAN=S(I,JJ)-AINT(S(I,JJ))
      IF(REMAN.GE.0.9999995) GO TO 201
      IF(REMAN.LE.0.0000005) S(I,JJ)=AINT(S(I,JJ))
      IF(REDMAN.LE.FUO) GO TO 202
      FUO=REMAN
      LL=I
      GO TO 202
  201 S(I,JJ)=S(I,JJ)+0.0000005
      S(I,JJ)+AINT(S(I,JJ))
  202 CONTINUE
      IF(FUO.LE.0.5E-5) GO TO 1
```

```
C
  300  INTS=JJ-II
       DO 216 I=1,INTS
  216  DUJ(I)=0.
       JC=1
       DO 218 I=1,JJ
       DO 220 J=2,II
  220  IF(I.EQ.L(J)) GO TO 218
       LD(JC)=I
       JC=JC+1
  218  CONTINUE
       JC=JC-1
       DO 212 J=1,JC
       NOTB=LD(J)
       IF(S(LL,NOTB).LT.0.)GO TO 213
       FUJ=S(LL,NOTB)-AINT(S(LL,NOTB))
       GO TO 212
  213  ABSY=ABS(S(LL,NOTB))
       FUJ=ABSY-AINT(ABSY)
       FUJ=1.-FUJ
  212  DUJ(J)=FUJ
C      INSERT THE NEW CUT INTO MATRIX
       JJJ=JJ+1
       DO 226 I=1,II
  226  S(I,JJJ)=S(I,JJ)
       DO 228 I=1,III
  228  S(I,JJ)=0.
       S(II1,JJ)=1.
       S(III,JJJ)=-FUO
       DO 230 I=1,INTS
       NEWCUT=LD(I)
  230  S(III,NEWCUT)=-DUJ(I)
       L(III)=JJ
       JJJ=JJ
       JJ=JJ+1
       II=II+1
       III=II+1
       IF(III.EQ.50) GO TO 1
       DO 232 I=1,JJ
  232  S(III,I)=0.
       REGA=1.
       GO TO 21
C      MIN XBI
  500  JK=0
       IF(REGB.EQ.0.) GO TO 62
       XMIN=0.
```

```
      DO 302 I=2,II
      IF(S(I,JJ).GE.XMIN) GO TO 302
      XMIN=S(I,JJ)
      JK=I
302   CONTINUE
      IF(XMIN.GE.0.) GO TO 70
      IF(XMIN.LE.05E-4) GO TO 70
      KJ=0
      XMAX=-0.5E12
      DD 304 J=1,JJJ
      IF(S(JK,J).GE.0.) GO TO 304
      X=S(1,J)/S(JK,J)
      IF(X.LE.XMAX) GO TO 304
306   XMAX=X
      KJ=J
304   CONTINUE
      IF(KJ.EQ.0) GO TO 1
      GO TO 56
  1   WRITE(6,100)
100   FORMAT(////' THE FINAL MATRIX')
      DO 78 I=1,III
      WRITE(6,150) I
150   FORMAT(//35X,'ROW ',I2/,1X)
 78   WRITE(6,12) (S(I,J),J=1,JJ)
      WRITE(6,7)
  7   FORMAT (1X,'VARIABLE          VALUE')
      DO 71 I=2,II
 71   WRITE (6,5) L(I),S(I,JJ)
  5   FORMAT (I4,F20.8)
      STOP
      END
```

```
C       THIS PROGRAM USE THE ALGORITHM SUGGESTED BY
C       DR. FERYDOOM KIANFAR
C       FOR SOLVING 0,1 ALL INTEGER PROGRAMMING PRO-
C       BLEMS USING KNAPSACK
C       FUNCTIONS. WHERE II=TOTAL NUMBER OF ROWS.
C       JJ=TOTAL NUMBER OF COLUMNS. S(I,J)=TABLEAU FOR
C       SIMPLEX METHOD.
C       L(I)=VARIABLES IN THE BASIS.
C       B(I)=THE PIVOT ROW.
C       LD(I)=VARIABLES NOT IN THE BASIS.
C       FT(I)=THE F VECTOR.
C       SS(I,J)=THE INITIAL PRIMAL FEASIBLE TABLEAU.
C       P(I,I)=THE P MATRIX.
C       FP(I)=THE VECTOR COME FROM P MATRIX MULTIPLY
C       BY FT VECTOR.
C       IDA(I)=TABLEAU CONTAINING THE VALUE OF D.
C       IDB(I)=TABLEAU CONTAINING THE VALUE OF D NOT IN IDA.
C       IARH(I)=NEW VALUE OF A.
C       IDRR=INDEX VALUE FOR TABREAU IDA(I) AND IDB(I).
C       IFP(I)=INTEGER VALUE OF FP(I).
C       NARI(I)=THE LEGITIMATE VARIABLES.
        DIMENSION S(46,96),  B(46), L(46),  FT(30),  LD(30)
        DIMENSION IDB(300), IDRR(30), FP(30), NARI(15)
        DIMENSION SS(46,96), P(30,30), IFP (30), IDA(300), IARH(30)
        READ(5,4) II, JJ
    4   FORMAT (18I4)
        III=II+1
C       READ IN THE MATRIX
        DO 10 I=1, III
   10   READ(5,12) (S(I,J), J=1, JJ)
   12   FORMAT(7F10.4)
        DO 3 I=1, III
        DO 3 J=1, JJ
    3   SS(I,J)=S(I,J)
C       READ IN THE SUBSCRIPT FOR THE SLACK VARIABLE
        READ(5,4) (L(I), I=2, II)
C       READ IN VARIABLES THAT NEED TO BE INTEGER
        READ(5,4) NN, (NARI(I), I=1, NN)
        IIII=II-1
        REGA=0.
   21   KKK=0
   22   DO 23 I=2, II
        IF(L(I). NE. 0) GO TO 23
        DO 27 J=1, JJ
        IF(S(I,J). EQ. 0. ) GO TO 27
        S(III,J)=S(III,J)-S(I,J)
```

```
   27  CONTINUE
   23  CONTINUE
C      SEARCHING FOR THE MOST NEGATIVE ENTRY
       K=III
   44  JJJ=JJ-1
       IF(REGA.EQ.1.) GO TO 500
       B(K)=0.
       L(K)=0.
       DO 42 J=1,JJJ
       IF(S(K,J).GE.0.) GO TO 42
       IF(B(K).LE.S(K,J)) GO TO 42
       B(K)=S(K,J)
       L(K)=J
   42  CONTINUE
C      TEST FOR L(K).
   45  IF(L(K)) 46,62,46
C      FIND OUT THE PIVOT COLUMN
   46  KJ=L(K)
       DO 120 I=2,II
       IF(S(I,KJ)) 120,120,121
  120  CONTINUE
       WRITE(6,130)

  130  FORMAT(1X,'UNBOUNDED')
       GO TO 1
C      COMPUTING THE RATIO
  121  JK=0
       DO 50 I=2,II
       IF(S(I,KJ).LE.0.) GO TO 50
       X=S(I,JJ)/S(I,KJ)
       IF(JK.EQ.0) GO TO 53
       IF(X.GE.XMIN) GO TO 50
   53  XMIN=X
       JK=I
   50  CONTINUE
   56  X=S(JK,KJ)
       WRITE(6,149) X
  149  FORMAT(/,' X= ',F14.6)
       L(JK)=KJ
C      CALCULATING THE NEW ROWS ABOVE THE PIVOT ROW
       DO 57 I=1,III
   57  B(I)=S(I,KJ)
       IJ=JK-1
       DO 59 I=1,IJ
       DO 59 J=1,JJ
       IF(S(JK,J)) 58,59,58
```

```
   58  IF (B(I))580, 59, 580
  580  S(I, J)=S( I, J) - B(I)*(S(JK, J)/X)
   59  CONTINUE
C      CALCULATING THE NEW ROWS BELOW THE PIVOT ROW
       IJ=JK+1
       DO 61 I=IJ, III
       DO 61 J=1, JJ
       IF (S(JK, J)) 60, 61, 60
   60  IF (B(I))600, 61, 600
  600  S(I, J)=S(I, J)--B(I)*(S(JK, J)/X)
   61  CONTINUE
C      NORMALIZATION
       DO 205 J=1, JJ
  205  S(JK, J)=S(JK, J)/X
       KKK=KKK+1
       WRITE(6, 105) KKK, S(1, JJ), L(JK)
  105  FORMAT (1X, I4, 6X, F15. 2, 10X, I4)
       GO TO 44
   62  IF(K-1) 70, 70, 63
   63  IJ=JJ-1
C      TEST TO SEE WHETHER ALL THE ELEMENTS ON THE LAST ROW
C      ARE CLOSE TO ZERO.
       DO 65 J=1, IJ
       IF(S(K, J)-. 0001) 65, 65, 66
   65  CONTINUE
       WRITE (6, 103)
  103  FORMAT (1X, 'FEASIBLE')
       WRITE(6, 101)
  101  FORMAT(1X, 'ITERATION      OBJ.FUNCTION      NEW BASIC VAR.
C      TEST TO SEE WHETHER ALL THE ELEMENTS ON THE LAST ROW
       DO 140 J=1, JJ
  140  S(III, J)=0.
       K=1
       KKK=0
       GO TO 44
   66  WRITE(6, 6)
    6  FORMAT(1X, 'INFEASIBLE')
       GO TO 1
   70  WRITE(6, 8) S(1, JJ)
    8  FORMAT(////1X, '0BJ. FUNCTION. ', F20. 8/, 1X)
       WRITE(6, 7)
    7  FORMAT (1X, 'VARIABLE          VALUE')
       DO 71 I=2, II
   71  WRITE (6, 5) L(I), S(I, JJ)
    5  FORMAT(I4, F20. 8)
C      FIND THE LARGEST S(I, JJ)
```

```
      FUO=0.
      DO 202 I=2,II
      REMAN=S(I,JJ)-AINT(S(I,JJ))
      IF (REMAN.GE. 0.9999995) GO TO 201
      IF (REMAN.LE. 0.0000005) S(I,JJ)=AINT(S(I,JJ))
      IF REMAN.LE. FUO) GO TO 202
      FUO=REMAN
      LL=I
      GO TO 202
  201 S(I,JJ)=S(I,JJ)+0,0000005
      S(I,JJ)=AINT(S(I,JJ))
  202 CONTINUE
C     TEST IF OPTIMAL SOLUTION HAS BEEN OBTAINED
      IF(FUO.LE. 0.5E-5) GO TO 1
      WRITE(6,954) JJ,FUO,LL
  954 FORMAT(1X,'JJ=',I6,'  FUO=',F10.4,'  LL=',I6)
  300 INTS=JJ-II
      DO 216 I=1,INTS
  216 FT(I)=0.
      JC=1
      DO 218 I=1,JJ
      DO 220 J=2,II
  220 IF(I.EQ.L(J)) GO TO 218
      LD(JC)=I
      JC=JC+1
  218 CONTINUE
      JC=JC-1
      INTSS=INTS+1
      P(1,1)=1.
      DO 502 J=2.INTSS
  502 P(1,J)=0.
      DO 504 J=1,JC
      NOTB=LD(J)
      IF(NOTB.LE. NN) GO TO 503
      DO 506 I=2,II
  506 IF(SS(I,NOTB).EQ.1.) GO TO 508
      WRITE(6,510)
  510 FORMAT(/' ERROR IN S MATRIX')
  508 JT=J+1
      DO 512 K=1,INTS
      KT=K+1
  512 P(JT,KT)=-SS(I,K)
      P(JT,1)=SS(I,JJ)
      GO TO 504
  503 NOT J=J+1
      NOTB=NOT B+1
```

```
      DO 505 MI=1,INTSS
  505 P(NOTJ,MI)=0.
      P(NOTJ,NOTB)=1.
  504 CONTINUE
C     COMPUTE FXP
      FT(1)=FUO
      DO 517 I=2,INTSS
      IT=I-1
      IF(S(LL,LD(IT)).LT.0.) GO TO 513
      REMAN=S(LL,LD(IT))-AINT(S(LL,LD(IT)))
      GO TO 516
  513 ABSY=ABS(S(LL,LD(IT)))
      REMAN=ABSY-AINT(ABSY)
      REMAN=1.-REMAN
  516 FT(I)=-REMAN
  517 IF(ABS(FT(I)).GE.0.999995) FT(I)=0.
      WRITE(6,916)
  916 FORMAT ('  FT VECTOR')
      WRITE(6,12) (FT(I),I=1,INTSS)
C     FXP
      DO 518 I=1,INTSS
  518 FP(I)=0.
      DO 521 I=1,INTSS
      DO 520 J=1,INTSS
  520 FP(I)=P(J,I)*FT(J)+FP(I)
      IF(FP(I).GE.0.) IFP(I)=FP(I)+0.5
  521 IF(FP(I).LT.0.) IFP(I)=FP(I)-0.5
      LB=-(FP(1)-0.5)
      WRITE(6,950) LB,FB(1)
  950 FORMAT(1X,'LB=',I4,' FP(1)=',F10.4)
      WRITE(6,938)
  938 FORMAT('  FP(I)')
      WRITE(6,12) (FP(I),I=1,INTSS)
      WRITE(6,12) (IFP(I),I=1,INTSS)
      INR=INTSS
C     FIND D1 AND DJ, J=2,...,N
      IDA(1)=0
      IDA(2)=IFP(2)
      KC=1
      IDRR(1)=2
      K=3
      MM=2
      J=3
  538 I=1
  536 IF(I.GT.MM) GO TO 532
      NSTRAG=IDA(I)+IFP(J)
```

```
      IF(NSTRAG.GT.LB) GO TO 534
      IDP=K-1
      DO 535 ICC=1,IDP
  535 IF(NSTRAG.EQ.IDA(ICC))GO TO 534
      IDA(K)=NSTRAG
      K=K+1
  534 I=I+1
      GO TO 536
  532 MM=K-1
      J=J+1
      KC=KC+1
      IDRR(KC)=MM
      IF(J.NE.INTSS) GO TO 538
C     FIND AR HEAD
      KLA=LB-IFP(INTSS)
      KSNR=KLA-KPLAN(KLA,IDA,K)
      WRITE(6,918) KLA,LB,IFP(INTSS),KSNR
  918 FORMAT(2X,' KLA=  ',I4,'  LB=',I4,'  IFP(INTSS)=',I4,
     1 'KSNR=',I4)
      IN=INR-1
      IARH(IN)=IFP(INR)+KSNR
      IR=IN-1
  638 IF(IR.EQ.1) GO TO 603
      MINUSR=IR-1
      IPLUSR=IR+1
      IEADAK=LB-IFP(IPLUSR)
      DO 540 I=1,IDRR(MINUSR)
  540 IF(IEADAK.EQ.IDA(I)) GO TO 641
C     COMPUTE DRN AND FIND AR HEARD
      DO 542 I=1,IDRR(MINUSR)
  542 IDB(I)=IDA(I)
      I=1
      K=IDRR(MINUSR)+1
      MM=IDRR(MINUSR)
      INR=IN-IR
      CALL KNAP(INR,MM,K, IDB,IARH,LB,I,IPLUSR,IFP,&770,IR)
      GO TO 638
  641 IARH(IR)=IFP(IPLUSR)
      IR=IR-1
      GO TO 638
C     STEP 3
  603 IDB(1)=0
      MM=2
      K=3
      I=1
      IPLUSR=3
```

```
      IDB(2)=IARH(2)
      INR=IN-2
      CALL KNAP(INR, MM, K, IDB, IARH, LB, I, IPLUSR, IFP, &770, IR)
      GO TO 638
  770 DO 642 I=2, INTSS
      IOU=I-1
  642 FT(I)=IARH(IOU)
      FT(1)=FP(1)
C     GET P-1
      DO 704 I=1, NN
      IK=I+1
      DO 706 J=2, II
  706 IF(L(J). EQ. NARI(I)) GO TO 708
      DO 714 K=1, INTSS
  714 P(IK, K)=0.
      P(IK, IK)=1.
      GO TO 704
  708 DO 712 K=1, INTS
      KT=K+1
  712 P(IK, KT)=-S(J, LD(K))
      P(IK, 1)=S(J, JJ)
  704 CONTINUE
C     MATRIX FOP-1
      DO 718 I=1, INTSS
  718 FP(I)=0.
      DO 720 I=1, INTSS
      DO 720 J=1, INTSS
  720 FP(I)=P(J, I)*FT(J)+FP(I)
      WRITE (6, 930)
  930 FORMAT('  FOP-1')
      WRITE(6, 12) (FP(I), I=1, INTSS)
C     INSERT THE NEW CUT INTO MATRIX
      JJJ=JJ+1
      DO 226 I=1, II
      SS(I, JJJ)=SS(I, JJ)
  226 S(I, JJJ)=S(I, JJ)
      DO 228 I=1, III
      SS(I, JJ)=0.
  228 S(I, JJ)=0.
      S(III, JJ)=1.
      SS(III, JJ)=1.
      S(III, JJJ)=-FP(1)
      SS(III, JJJ)=-FT(1)
      DO 230 I=1, INTSS
      WRITE (6, 967) INTS, I, LD(I), FT(I)
  967 FORMAT(I4, I4, I4,  F16.8)
```

```
      NEWCUT=LD(I)
      IQ=I+1
      SS(III, I)=FT(IQ)
  230 S(III, NEWCUT)=FP(IQ)
      L(III)=JJ
      JJJ=JJ
      JJ=JJ+1
      II=II+1
      III=II+1
      IF(III.GE. 25) GO TO 1
      DO 232 I=1, JJ
  232 S(III, I)=0.
      REGA=1.
      GO TO 21
C     MIN XBI
  500 JK=0
      XMIN=0.
      DO 302 I=2, II
      IF(S(I, JJ). GE. XMIN) GO TO 302
      XMIN=S(I, JJ)
      JK=I
  302 CONTINUE
C     TEST XBR<0
      WRITE(6, 997) XMIN
  997 FORMAT(' XMIN=', F10. 6)
      IF(XMIN. GE. 0. ) GO TO 70
      KJ=0
      XMAX=-0. 5E12
      DO 304 J=1, JJJ
      IF(S(JK, J). GE. 0. ) GO TO 304
      X=S(1, J)/S(JK, J)
      IF(X, LE. XMAX) GO TO 304
  306 XMAX=X
      KJ=J
  304 CONTINUE
      IF(KJ. EQ. 0) GO TO 66
      GO TO 56
    1 WRITE(6, 100)
  100 FORMAT(////' THE FINAL MATRIX')
      STOP
      END
```

```
      SUBROUTINE KNAP(INR, MM, K, IDB, IARH, LB, I, IPLUSR, IFP, *, IR)
      DIMENSION IARH(30), IDB(300), IFP(30)
      DO 631 J=1, INR
  636 IF(I.GT. MM) GO TO 629
      NSTRBG=IDB(1)+IARH(IPLUSR)
      IF(NSTRBG, GT. LB) GO TO 634
      IDP=K-1
      DO 635 IDD=1, IDP
  635 IF(NSTRBG. EQ. IDB(IDD))GO TO 634
      IDB(K)=NSTRBG
      K=K+1
  634 I=I+1
      GO TO 636
  629 IPLUSR=IPLUSR+1
      MM=K-1
      I=1
  631 CONTINUE
      IPLUSR=IR+1
  632 KLA=LB-IFP(IPLUSR)
      KSNR=KLA-KPLAN(KLA, IDB, K)
      IARH(IR)=IFP(IPLUSR)+KSNR
      WRITE(6, 922) IR, IARH(IR)
  922 FORMAT(1X, 'IR=', I6, '  IARH(IR)=', I6)
      IR=IR-1
      IF(IR, EQ. 0) RETURN 1
      RETURN
      END



      FUNCTION KPLAN(KLA, IDA, K)
      DIMENSION IDA(300)
C     FIND KPLAN<DRN
      KPLAN=0
      MM=K-1
      DO 530 I=1, MM
      IF(IDA(I). LT. KPLAN. OR. IDA(I). GT. KLA) GO TO 530
      KPLAN=IDA(I)
  530 CONTINUE
      WRITE(6, 924) KPLAN
  924 FORMAT(1X, 'KPLAN=', I6)
      RETURN
      END
```